An Architecture Framework for Experimentations with Self-Adaptive Cyber-Physical Systems

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Michal Kit
Ilias Gerostathopoulos
Tomas Bures
Petr Hnetynka
Frantisek Plasil

iliasg@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE
faculty of mathematics and physics

Smart Cyber-Physical Systems

- Open-ended: no strict system boundaries
- Decentralized
- Physical world
  - Distribution
  - Mobility
- Communication
  - WiFi, 3G/4G, but also MANETS, VANETS etc.

How can we endow such systems with self-adaptive and self-organizing capabilities?

Department of Distributed and Dependable Systems

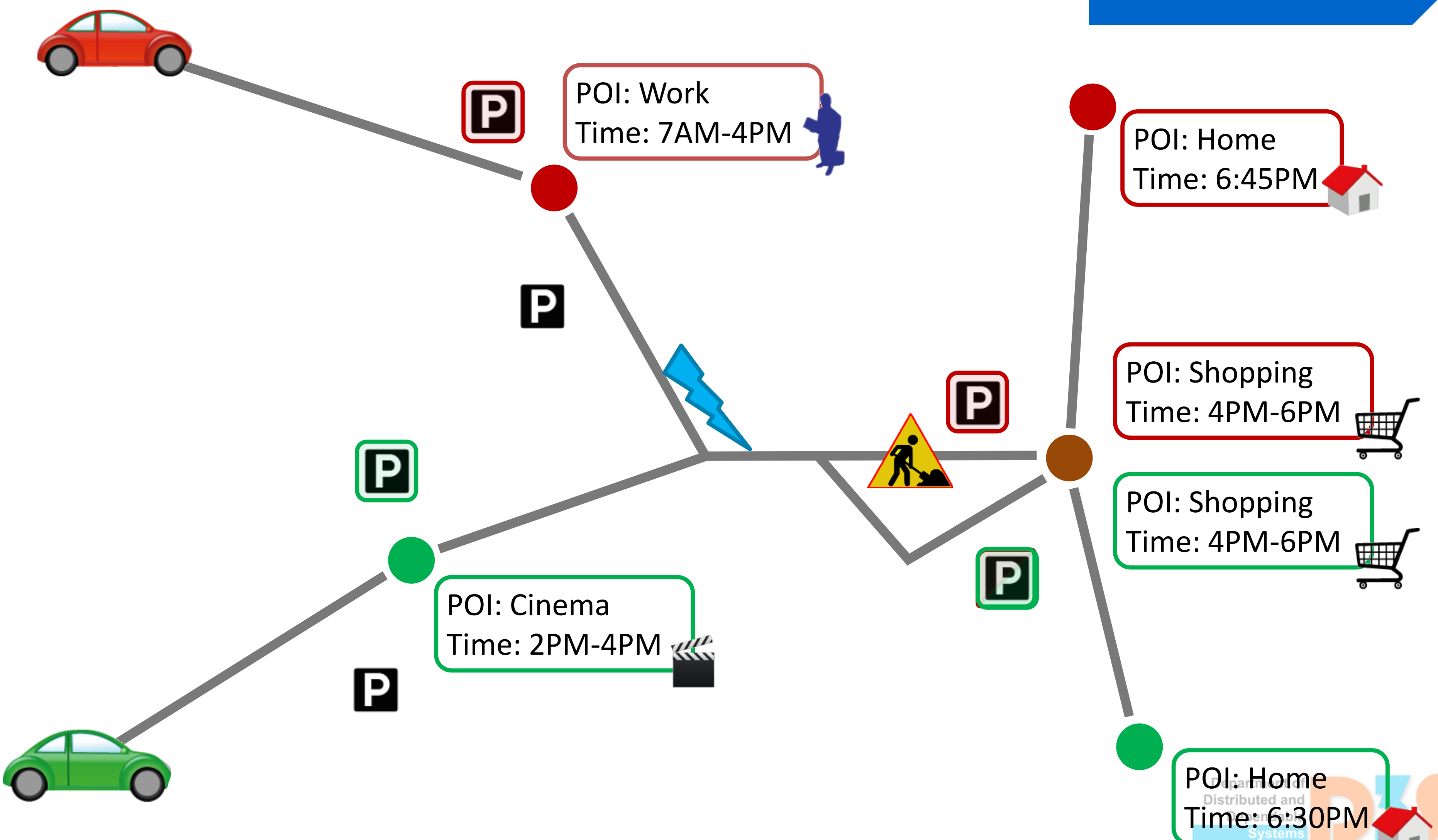
What is special about smart CPS?

Self-adaptation in smart CPS is hard
... especially when combined with dependability.

- **No global state**
- Dynamic physical structure
- Unstable connections: **no communication guarantees**
- **Communication delays**: data becomes obsolete
- Inherent dynamism stemming from external uncertainty and openness
- Emergent behavior

Department of Distributed and Dependable Systems

Example: E-mobility



Systematic Experimentation

To build self-adaptive smart CPS we need to first experiment with different adaptation approaches

An **experimentation framework** should have:

- Suitable **abstractions**
  - Goals, agents/components, component grouping
- **Simulation** capabilities
  - Network communication (including ad-hoc networks)
  - Environment behavior

Department of Distributed and Dependable Systems

DEECo

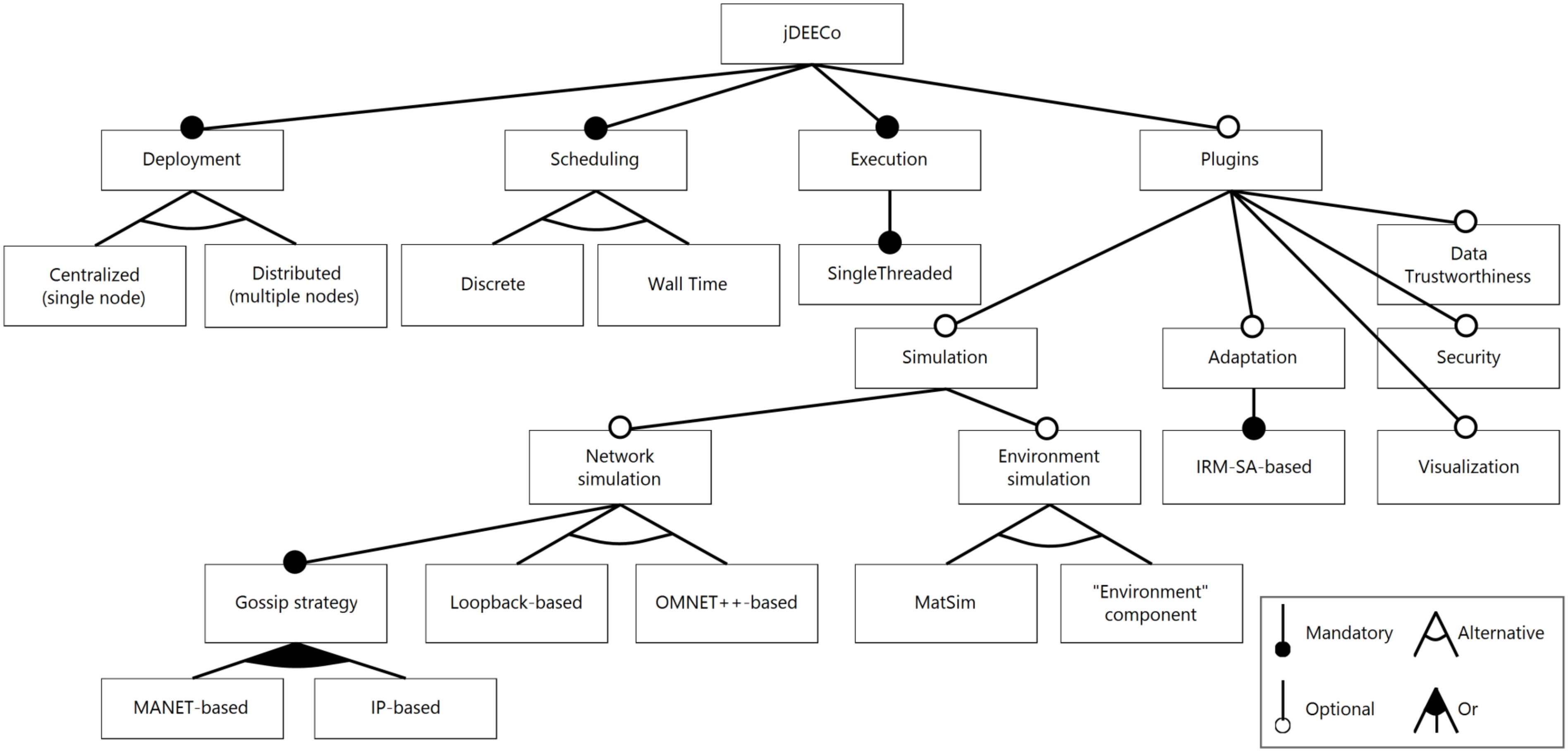
Component framework for self-adaptive smart CPS

- Suitable abstractions
  - Architecture: autonomous **components** & **ensembles**
  - Requirements: **invariants**
- Simulation capabilities
  - Network-accurate communication (OMNET++)
  - Agent-based simulations of the environment (MATSim)

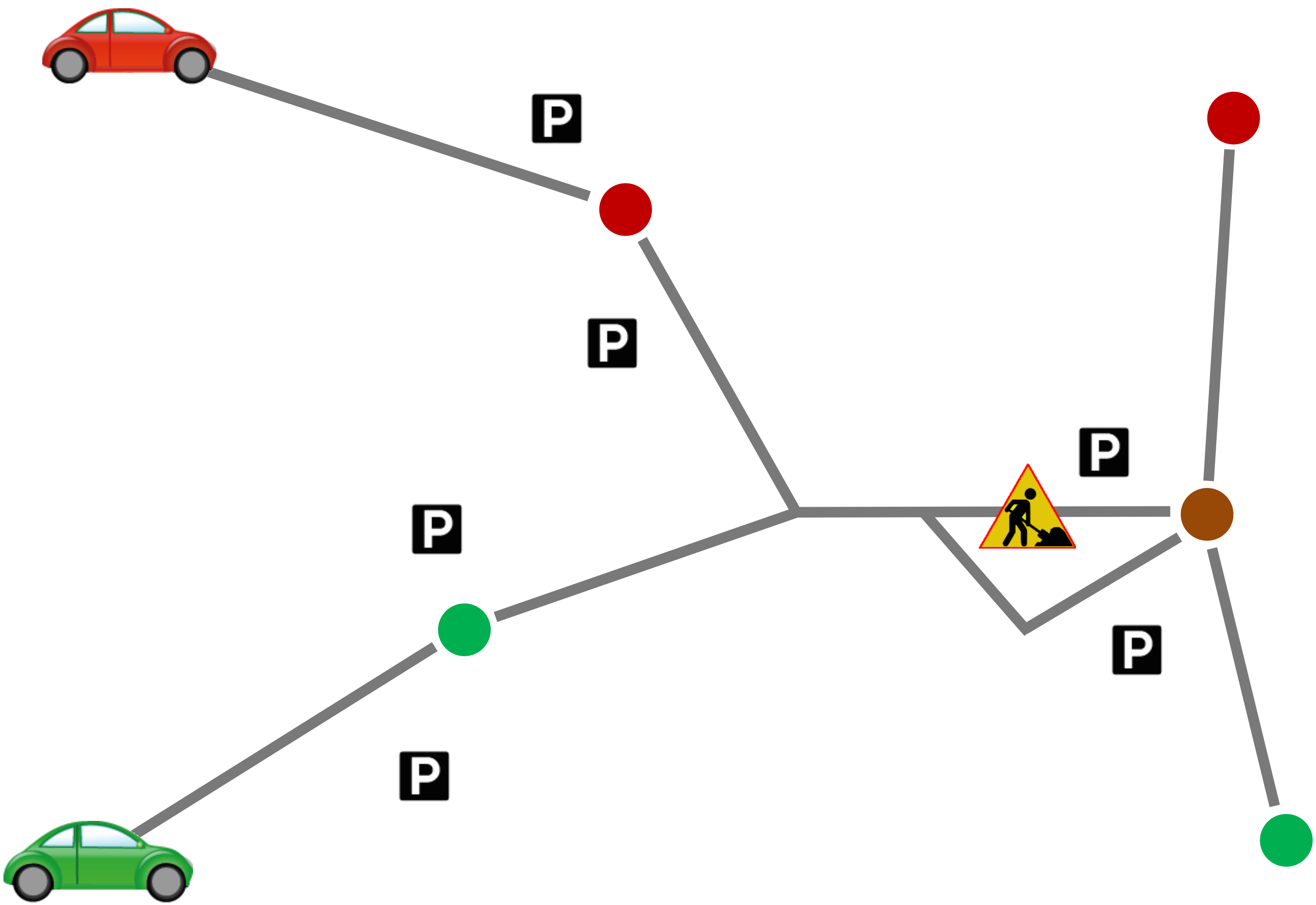
jDEECo: **Java-based** implementation of DEECo

- Based on internal Java DSL (Java annotations)

jDEECo Features

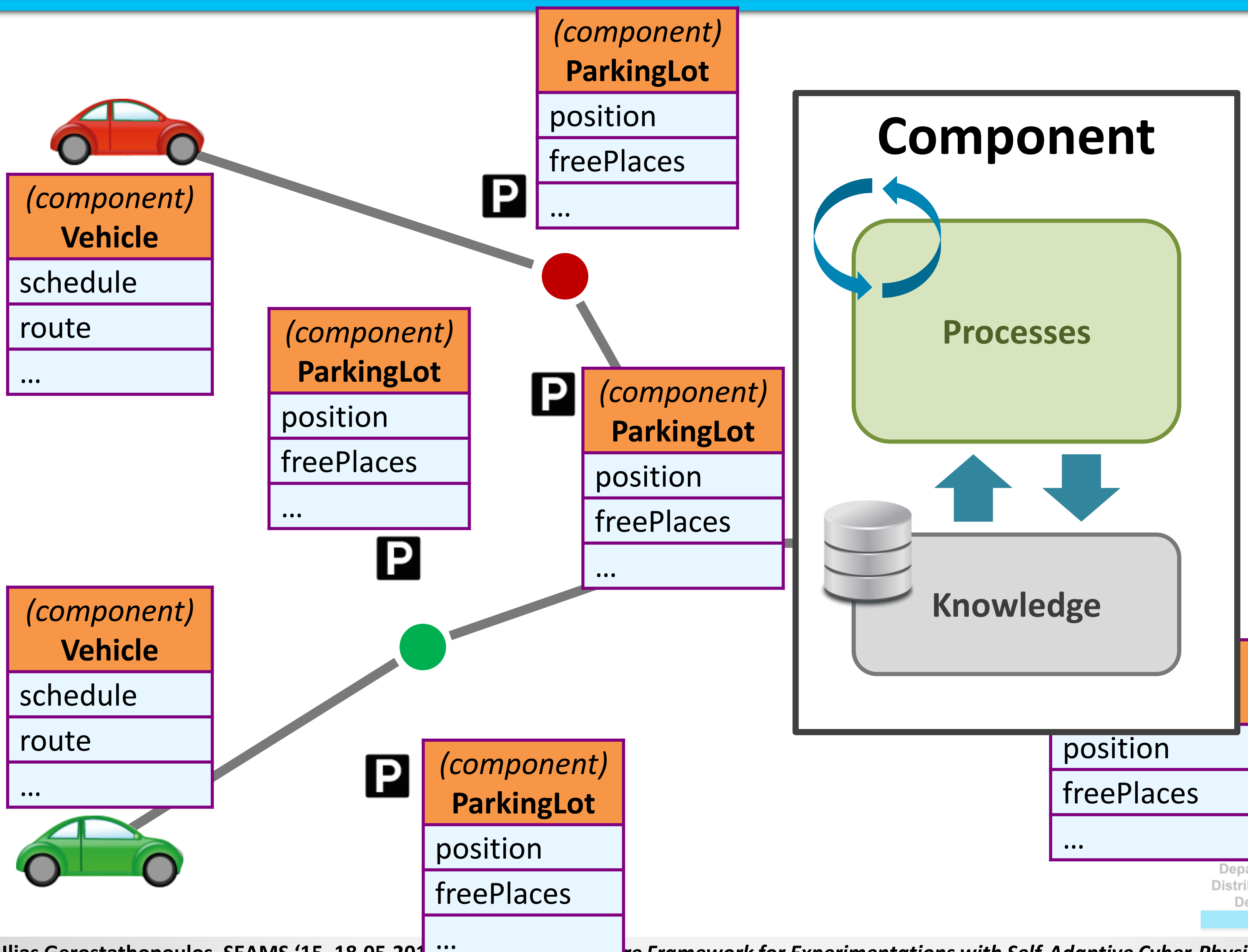


DEECo – Architecture Abstractions

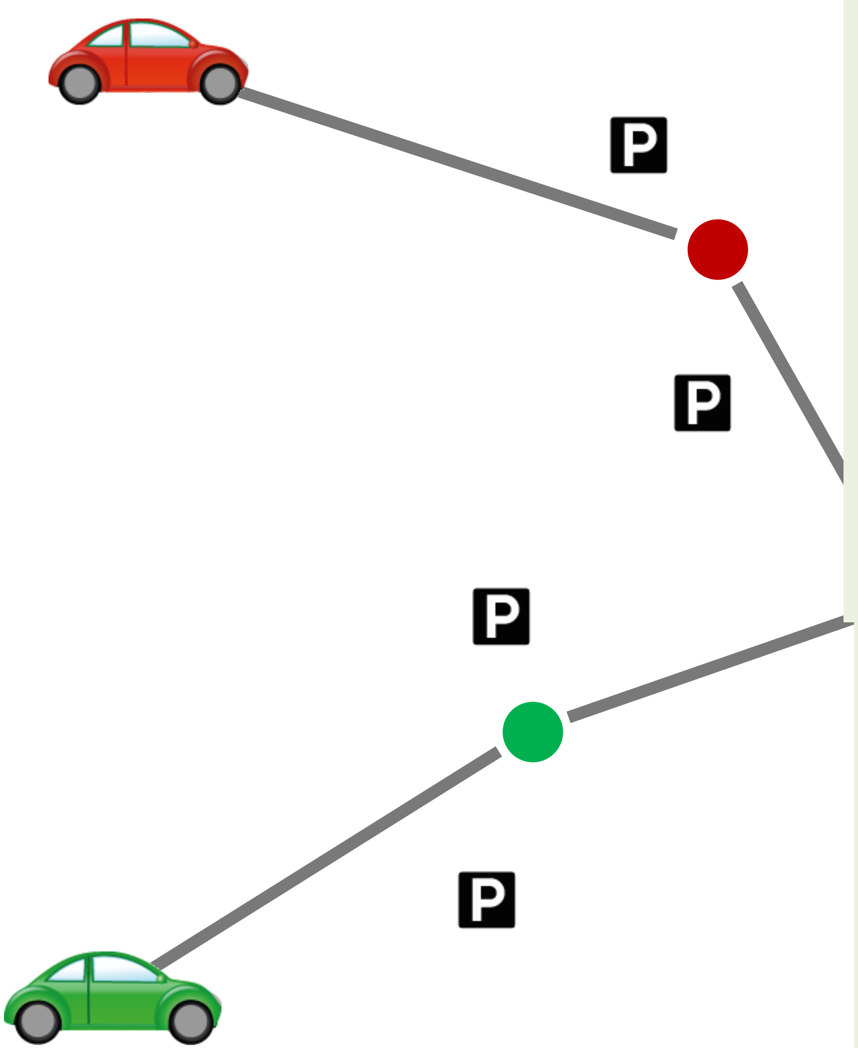


Department of
Distributed and
Dependable
Systems

DEECo – Architecture Abstractions

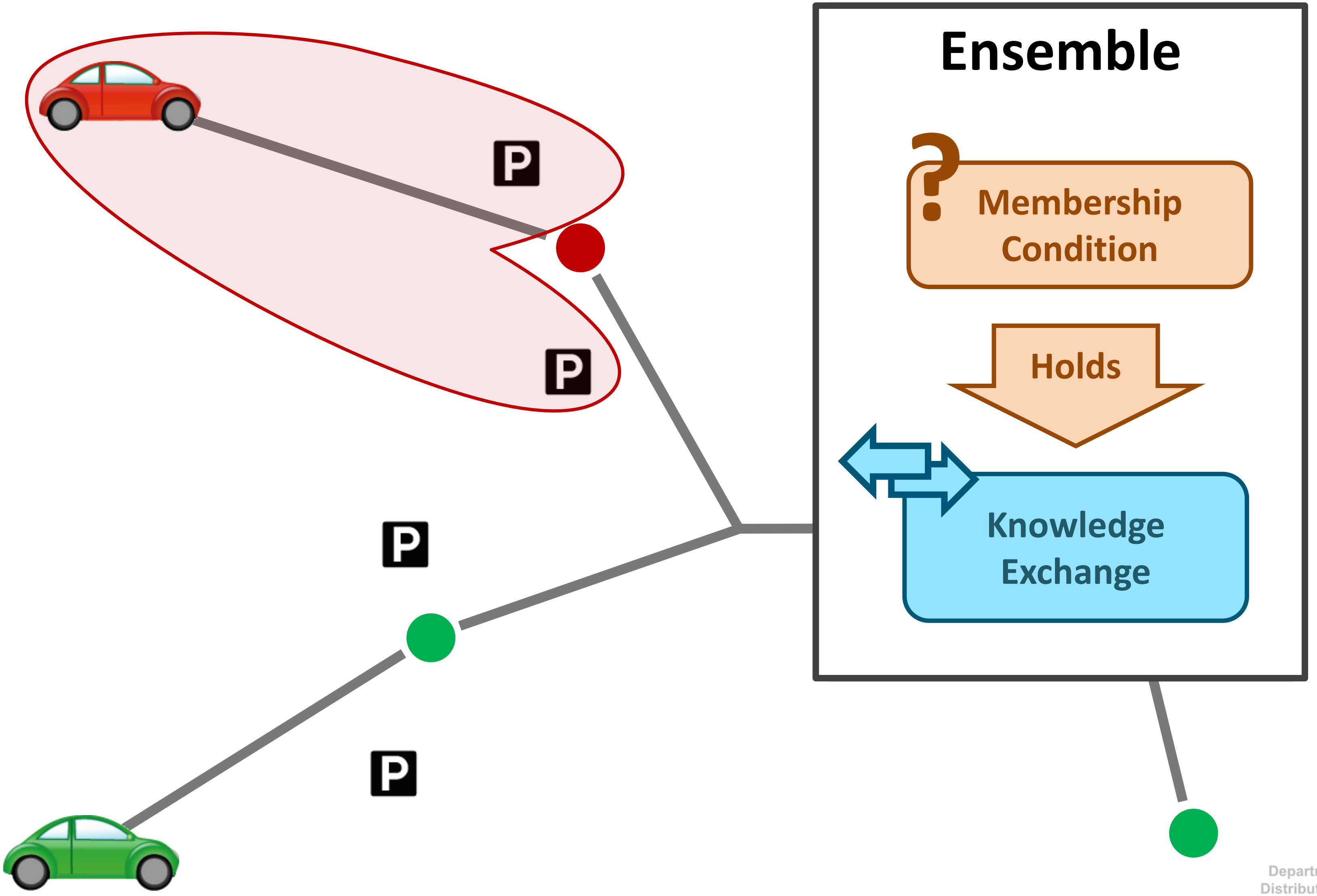


DEECo – Architectu

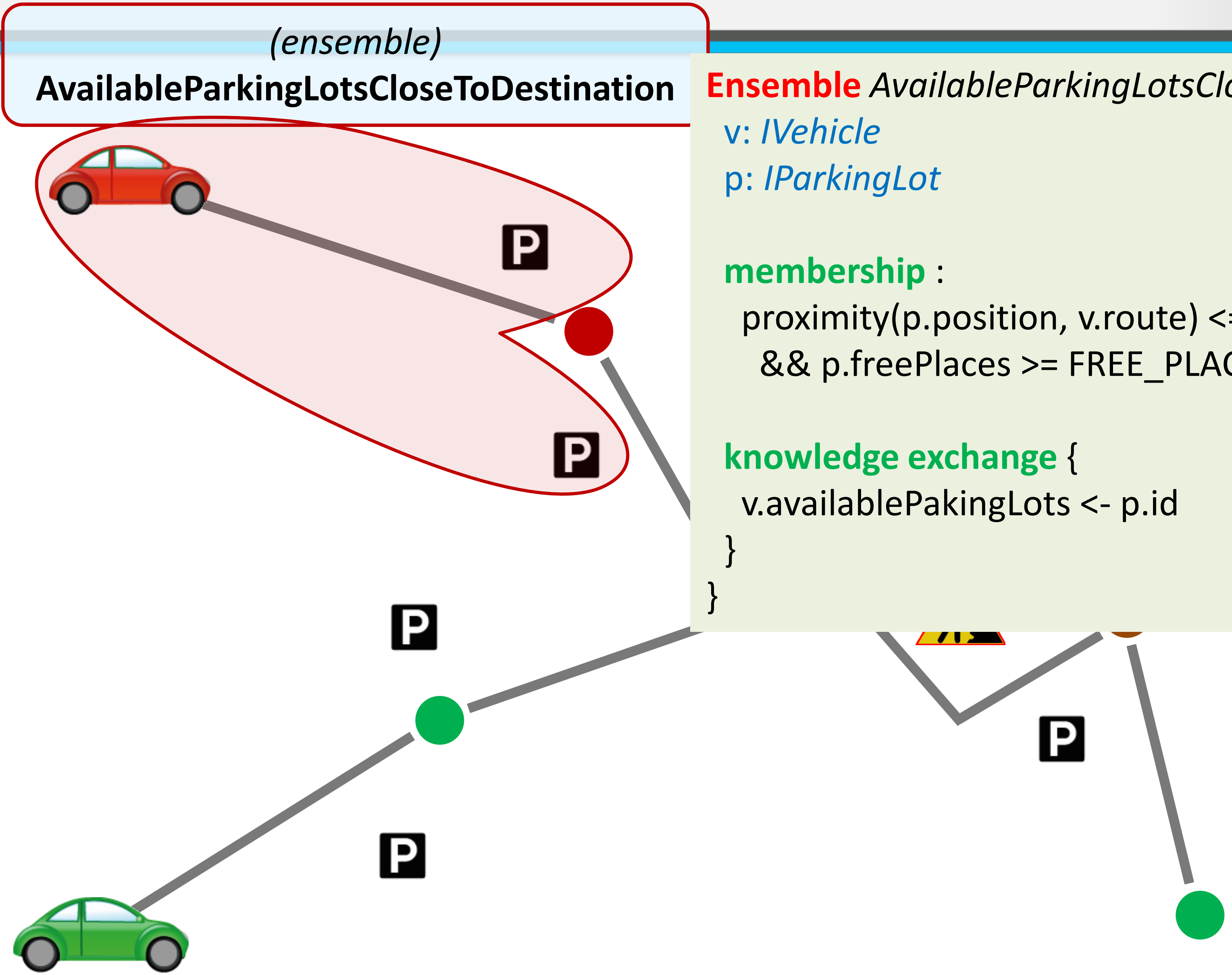



```
Component Vehicle = {
  position: IPosition
  availableParkingLots: IParkingLot[]
  route: IRoute
  schedule: ISchedule
  …
  process updatePlan {
    function = updatePlan
    inputKnowledge = [position,
        availableParkingLots, …]
    outputKnowledge = [route, …]
    scheduling = periodic(1s)
   …
} }

Component ParkingLot = {
  freePlaces: Int
  position: IPosition
  …
  process updateFreePlaces {
    …
  }
}
```

DEECo – Architecture Abstractions



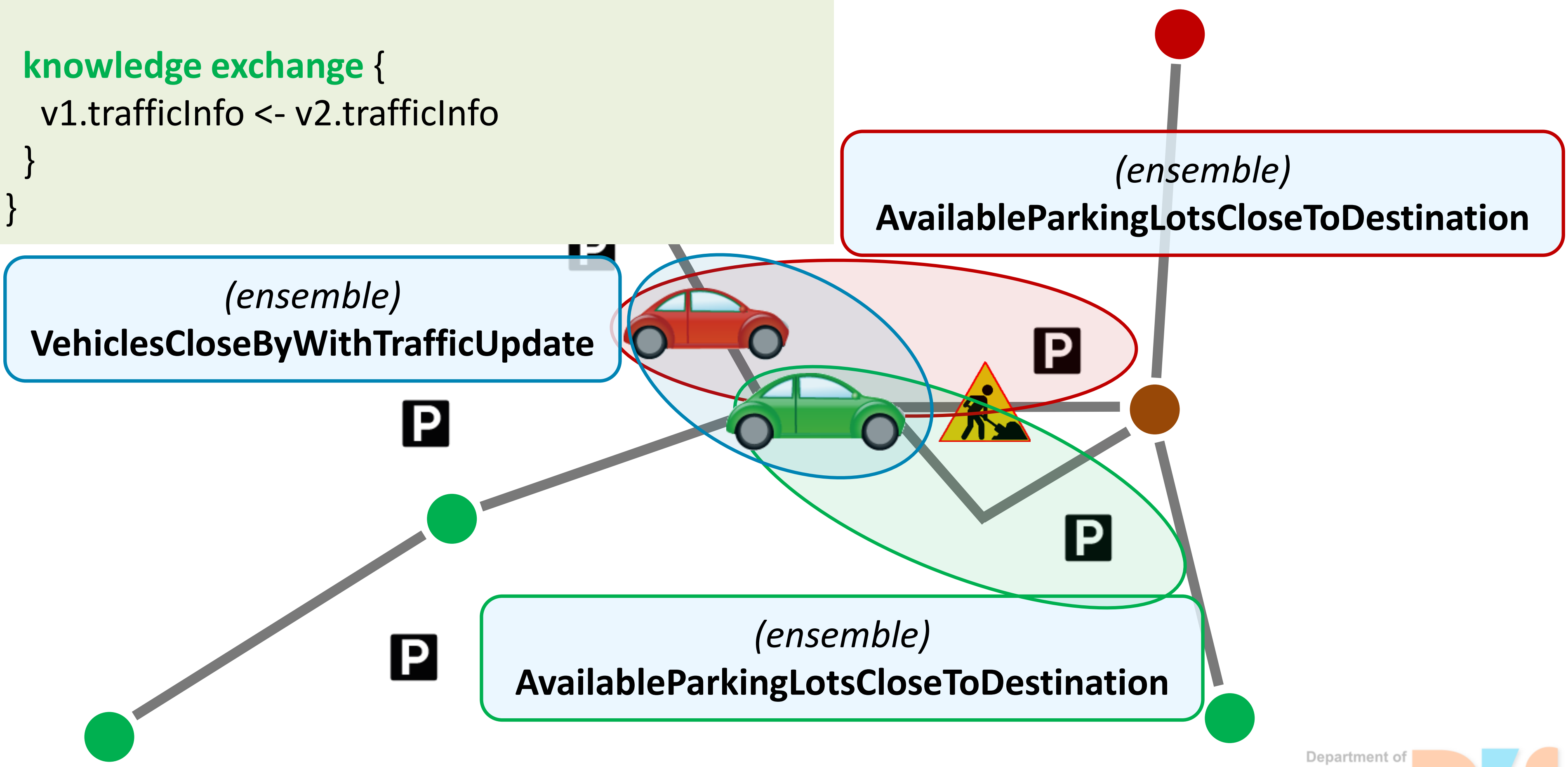
DEECo – Architecture Abstractions



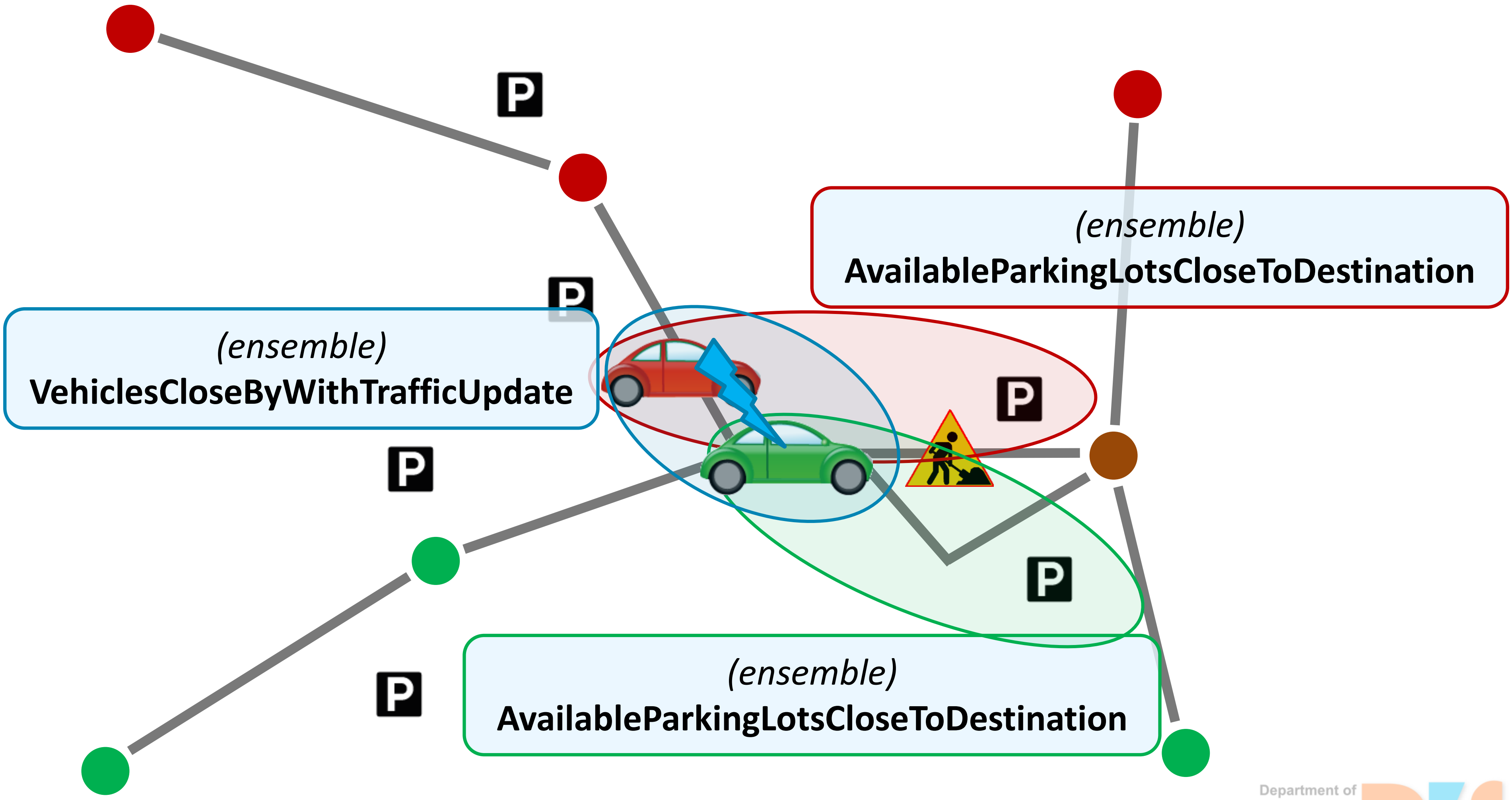

```
Ensemble AvailableParkingLotsCloseToDestination {
  v: IVehicle
  p: IParkingLot

  membership :
    proximity(p.position, v.route) <= DIST_THR
      && p.freePlaces >= FREE_PLACES_THR

  knowledge exchange {
    v.availablePakingLots <- p.id
  }
}
```

Abstractions

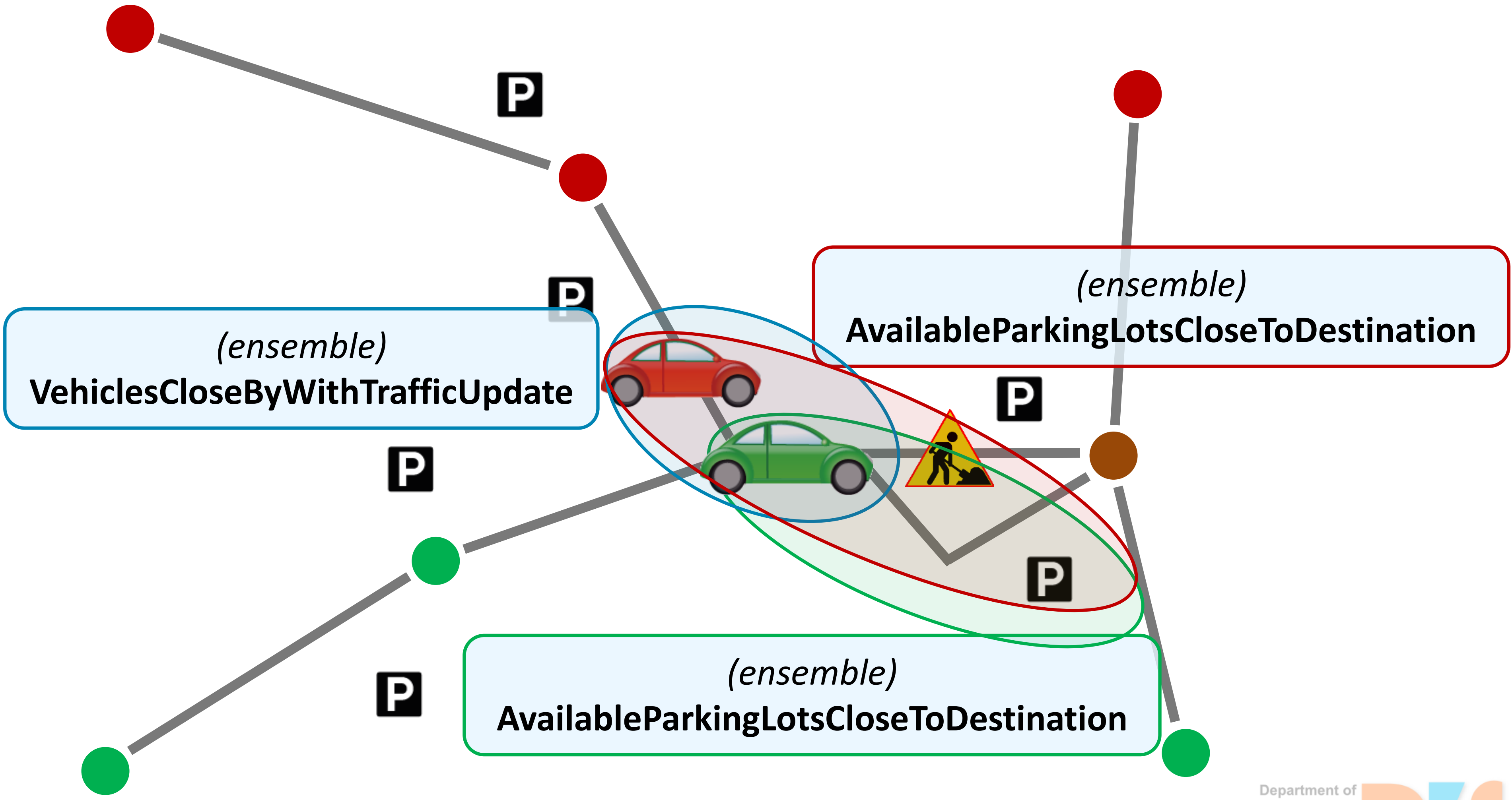
```
Ensemble VehiclesCloseByWithTrafficUpdate {
  v1: IVehicle
  v2: IVehicle

  membership :
    proximity(v1.position, v2.position) <= DIST_THR

  knowledge exchange {
    v1.trafficInfo <- v2.trafficInfo
  }
}
```



DEECo – Architecture Abstractions



DEECo – Architecture Abstractions



Requirements Abstractions

Invariants capture operational normalcy at every time instant

Decomposition of invariants forms design trees

- Akin to goal-oriented requirements elaboration

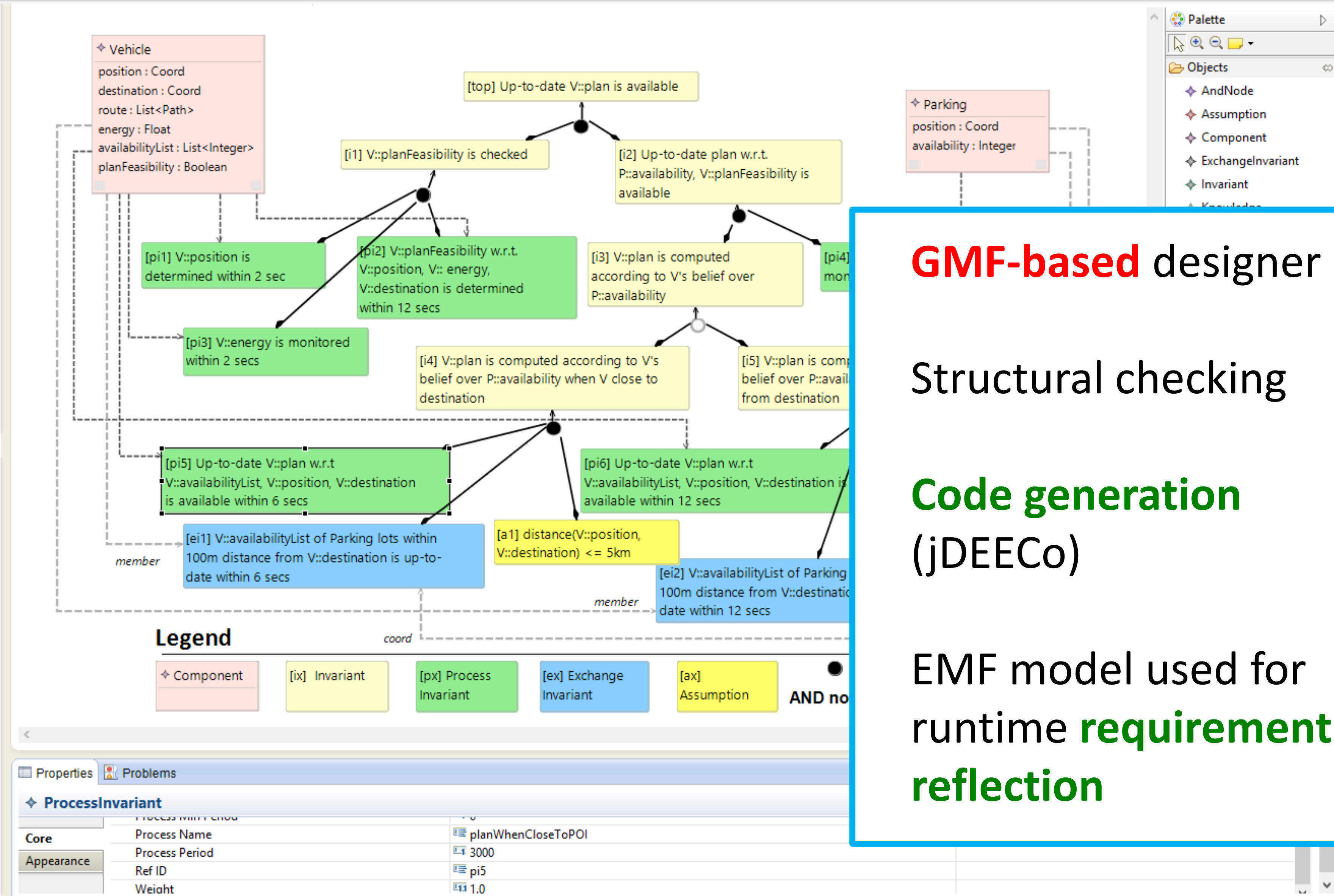
Leaf invariants are **operationalized** via

- Component processes and knowledge exchange functions
- Monitors

Alternative decompositions provide alternative system realizations

- Used at runtime to drive **architecture reconfiguration**

Support for Goal-Oriented Design



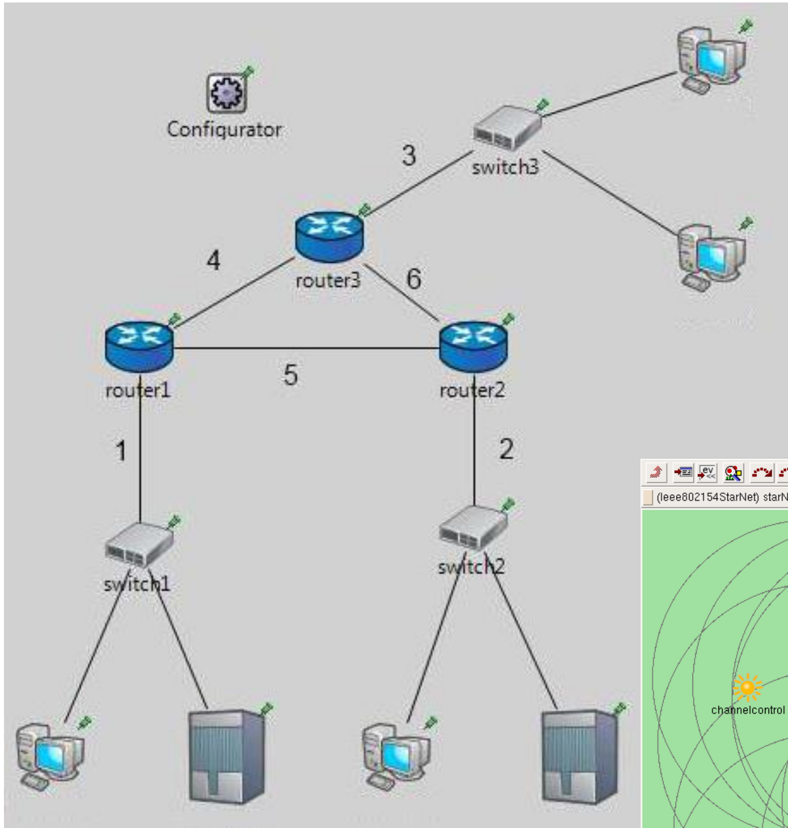
GMF-based designer

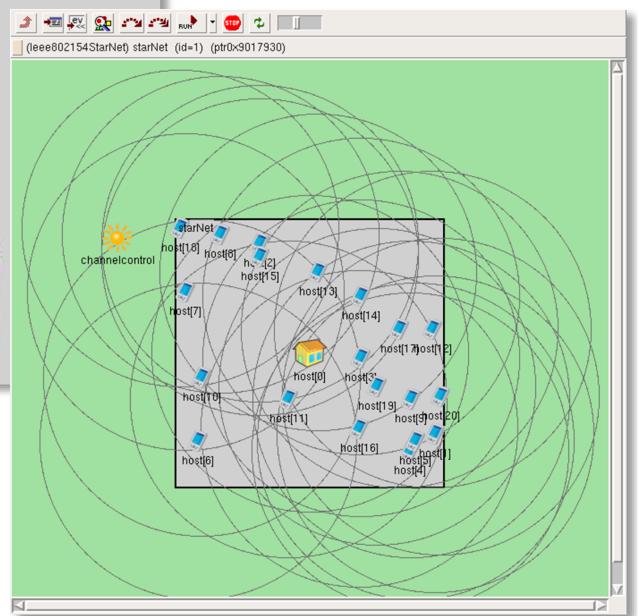
Structural checking

Code generation (jDEECo)

EMF model used for runtime **requirements reflection**

Support for Network Simulations





Detailed network simulator

Rich library of communication models

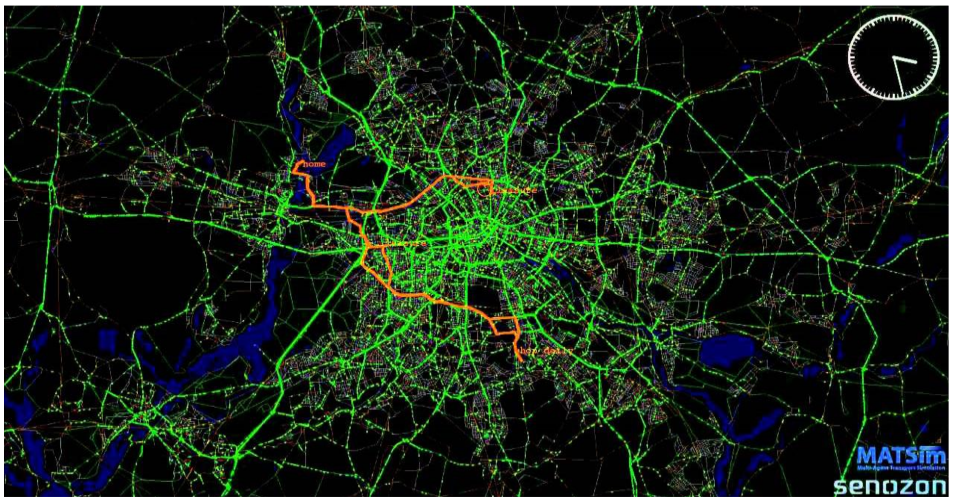
Rich library of hardware models

Support for Environment Simulations

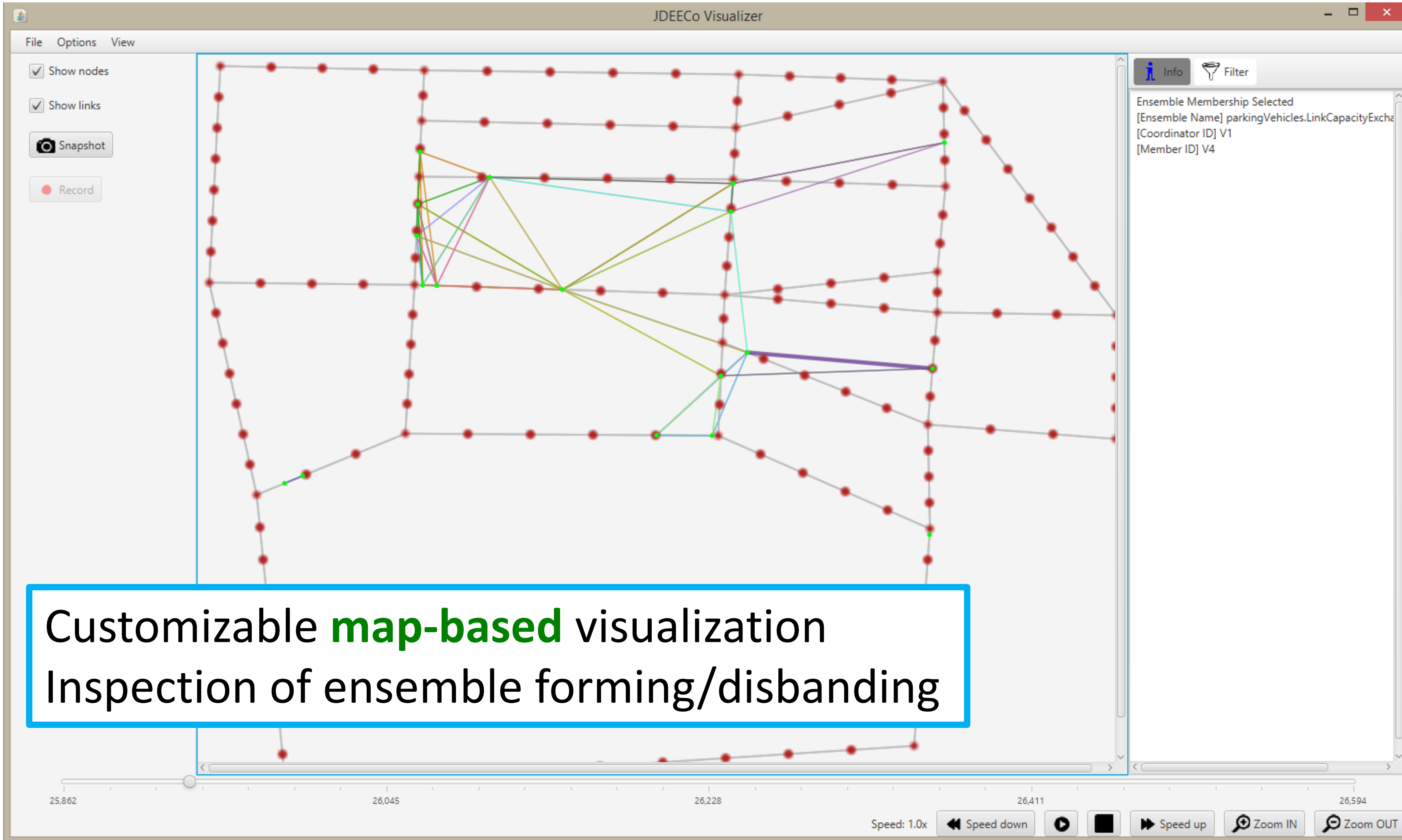
MATSim
Multi-Agent Transport Simulation

Agent-Based Traffic Simulator

Simulates people mobility according to their plans



Support for Visualization

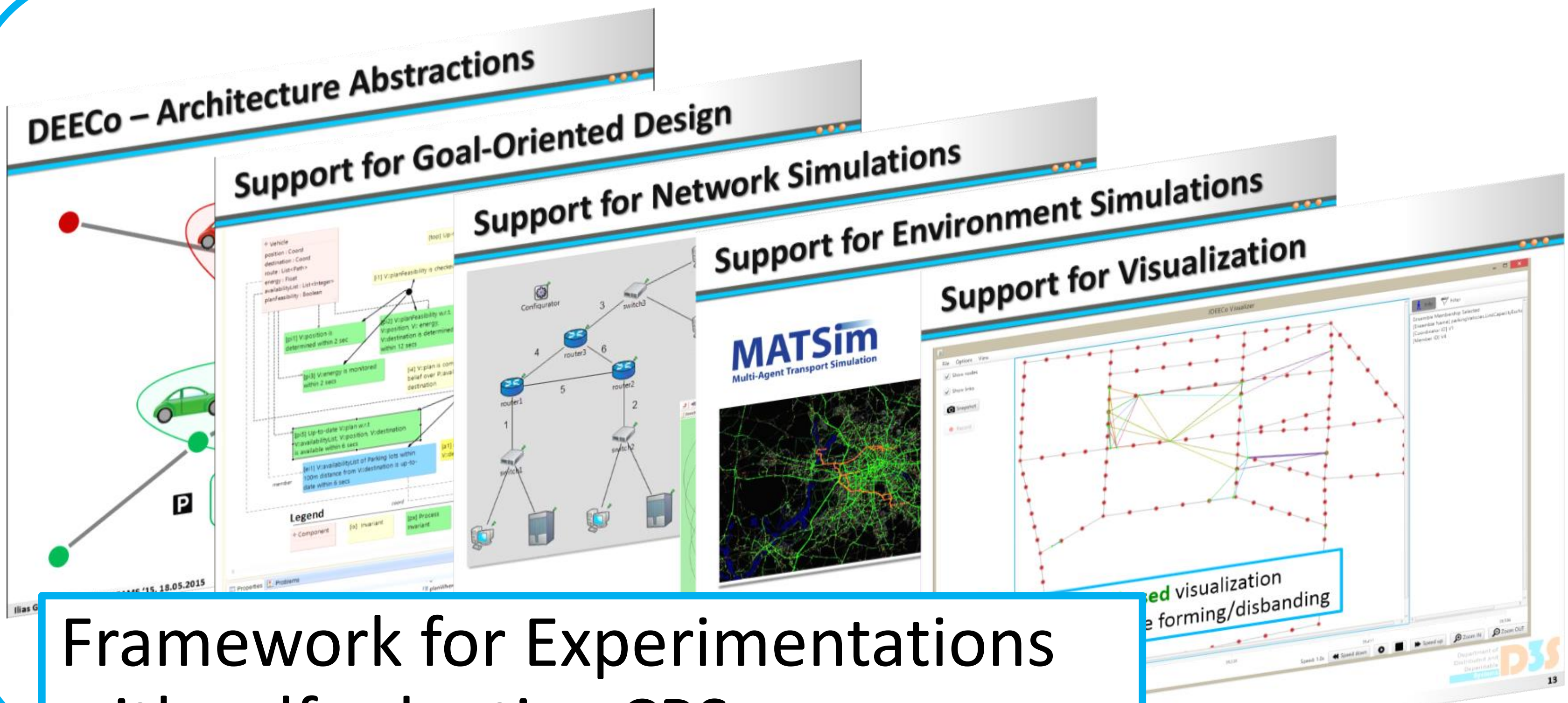


Customizable **map-based** visualization
Inspection of ensemble forming/disbanding

Department of Distributed and Dependable Systems

Conclusions

Self-adaptation is hard to achieve in smart CPS



Framework for Experimentations with self-adaptive CPS

https://github.com/d3scomp/JDEECo